Uniform Access to Time Series Data via the Heliophysics Application Programmer's Interface (HAPI)

Jon Vandegriff	JHU / APL
Todd King	UCLA
Jeremy Faden	Cottage Systems
Robert Weigel	GMU
Bernard Harris	GSFC
Aaron Roberts	GSFC
Nand Lal	GSFC
Robert Candey	GSFC

APL
JOHNS HOPKINS
APPLIED PHYSICS LABORATORY

Outline / Summary

A. motivation and context (HDMC effort)
B. data access API components and design goals (least common denominator)
C. the HAPI specification
   - status: version 1.1
   - REST-ful endpoints: `capabilities`, `catalog`, `info`, `data`
   - examples of endpoints
   - demo: in browser, IDL, Java / Autoplot
D. implementation scenarios: native versus pass-through
E. relationship of HAPI to SPASE (focus on data versus metadata)
F. who is using it (test servers at Goddard, U Iowa, GMU, PDS)
G. next steps if you want to use it (read the spec and talk to us)

https://github.com/hapi-server/data-specification

Terminology

parameter – a measured science quantity or a related ancillary quantity at one instant in time; may be scalar as a function of time, or a 1-D array at each time step; can have units and can have a fill value that represents no measurement or absent information

dataset – a collection with a conceptually uniform of set of parameters; one instance of all the parameters together with a time value constitutes a data record. Although a dataset is a logical entity, it often physically resides in one or more files that are accessible online. A HAPI service presents a dataset as a seamless collection of time ordered records, offering a way to retrieve the parameters while hiding actual storage details

request parameter – keywords that appear after the '?' in a URL with a GET request.

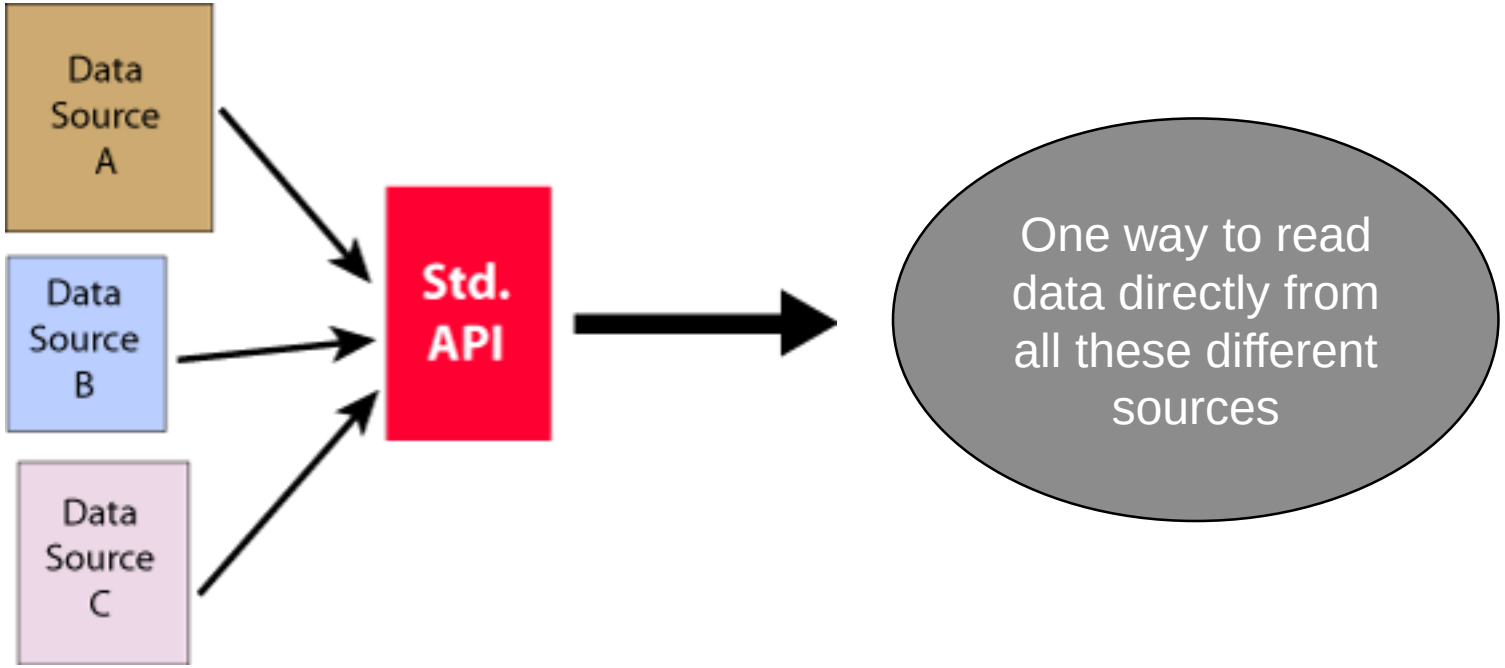
Consider this example GET request:

```
http://example.com/hapi/data?id=alpha&time.min=2016-07-13
```

The two request parameters are `id` and `time.min`. They are shown in bold and have values of `alpha` and `2016-07-13` respectively. This document will always use the full phrase "request parameter" to refer to these URL elements to draw a clear distinction from a parameter in a dataset.

Motivation (1 of 3): The Main Vision

To be able to write one piece of code that can access usable science data from anywhere using simple criteria.



Which API needs to be uniform?

The request API at a data center that programmers can use to get data. Sometimes referred to as the "fill my array" capability.

Many similar APIs exist. Consider:

- **CDAWeb has a web services API to get data**
- **U. Iowa has das2 servers**
- **LASP has LiZARD**
- **tsds.org (Weigel)**
- **the data request methods within IMPEx**

Motivation (2 of 3): Standardizing on the API rather than the format

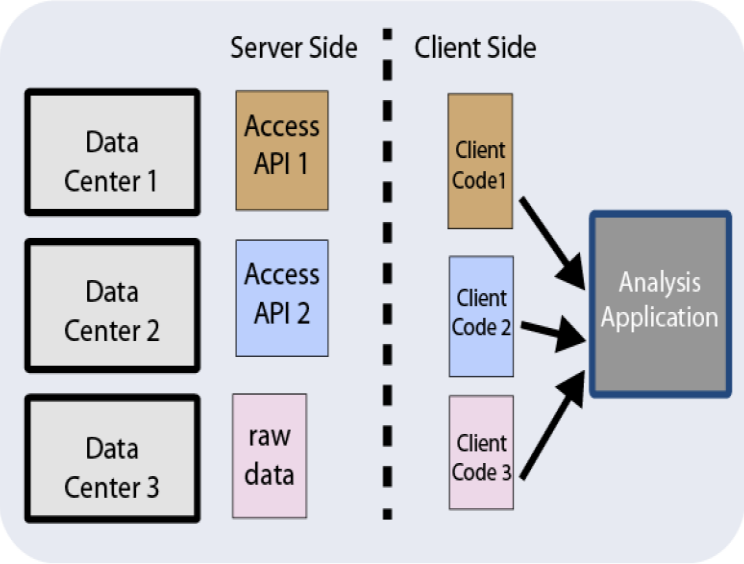
Past

Different data formats make interoperability difficult. ASCII (many flavors), CDF, HDF4, HDF5, netCDF, CEF, custom binary, etc.

There has been considerable progress in the use of standard formats recently, but there will never be just one data format.

Present

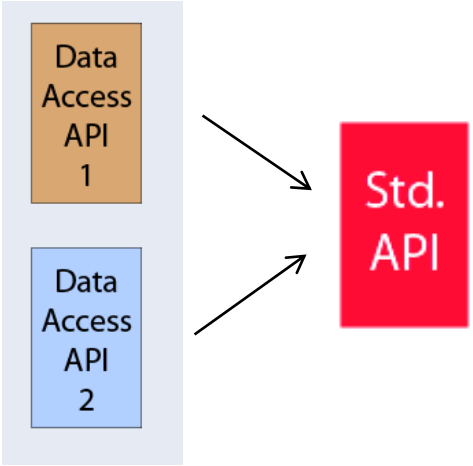
Data servers have consolidated many datasets, but each data center now has a different access interface.



Three access modes are shown here. The first two data centers have their own distinct APIs and the third data center just exposes a directory structure of files.

Future

need a domain-specific, but standardized API representing a lowest common denominator that multiple data centers will endorse and implement



note: also need to incorporate data centers without any existing API

Motivation(3 of 3): the need for something specific

Desired Criteria:

-data indexed primarily by time
-allows compact binary streaming format for high data volumes
-minimal effort for creating a server or a client
-can serve data of any native format

Existing server options / components:

OPeNDAP – too complex; not enough support for time series
VOTable – XML is too inefficient
Web Services – too generic – just an approach
Das2 Server – close, but stream format is too complex

None of these met all the criteria.

Improvements in HAPI over previous HDMC efforts:

simpler API with no bells and whistles
convergence of efforts by multiple groups

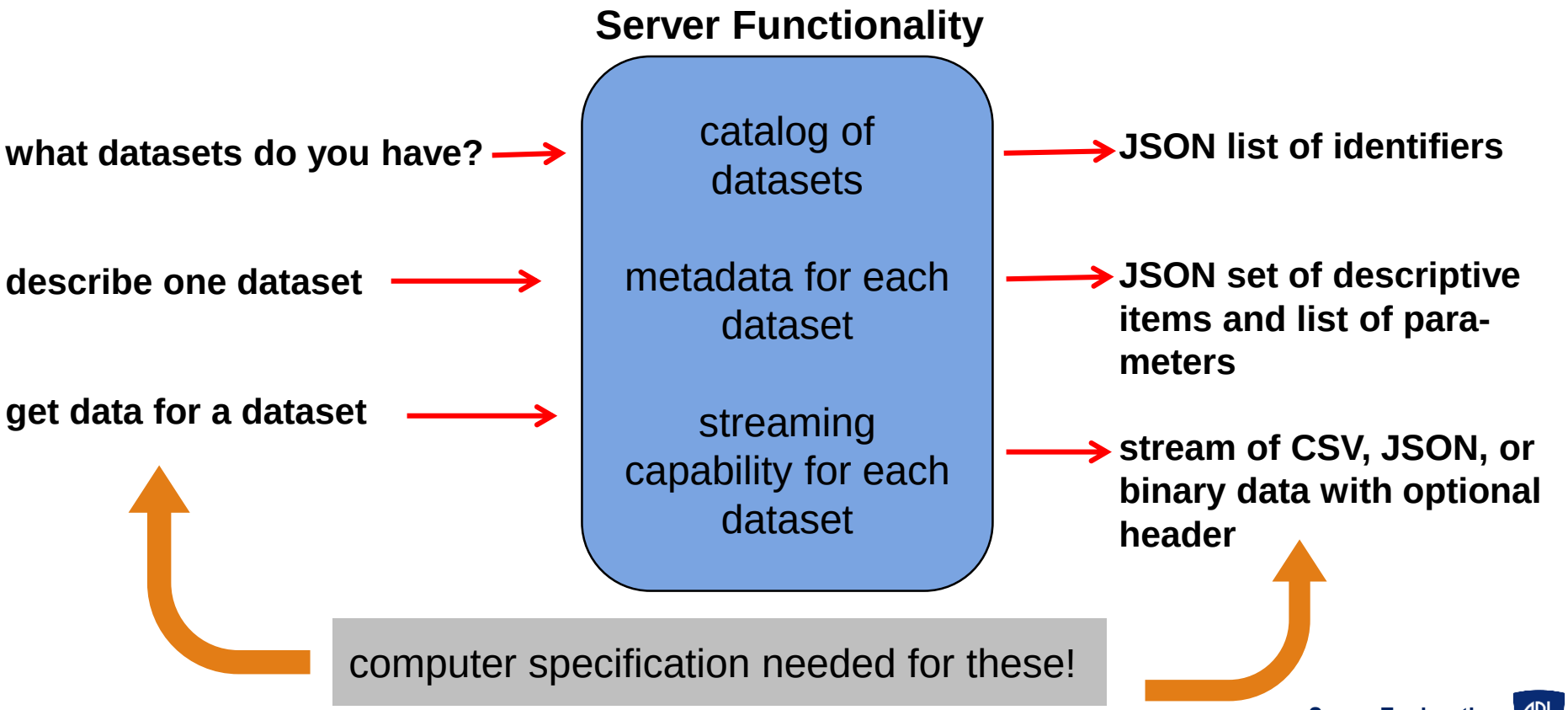
context:
NASA's Heliophysics Data & Model Consortium (HDMC)

- the HDMC houses the SPASE working group
- also supports a slow-burning effort to enhance the data environment within NASA's Heliophysics community
- overlaps informally with Planetary missions via plasma physics (energetic particles and fields)
- funding focus has been on
  - Heliophysics Virtual Observatories (VxOs)
  - data restoration
  - resident archives
  - data environment enhancements – a few funded efforts for new data serving ideas
- A group has been meeting for the last year to define a common data access API for a subset of Heliophysics data, namely time series datasets:

  Heliophysics Application Programmers Interface (HAPI)

Components of any access API

- **Two categories of components:**
  - **1. request format:** *how do you ask a server for data?*
  - **2. response format:** *what data format do you get in response*



HAPI Design Goals

- minimum set of server methods to expose one or more datasets on a server
- focus on data access, not data discovery
  - very little required metadata – only enough to identify the dataset and its parameters
  - independent of SPASE, which already handles the metadata
  - NOT intended to be a full interface for a complete data-center web site
- hide all native storage details of the data: present a conceptual view of a dataset as a list of parameters
- data retrieval through a stream format rather than files
- lowest common denominator request structure – no fancy features like filtering, merging, interpolation
- simple to implement in multiple languages
  - easy to write a basic **server** implementation from scratch
    (target = scientist who routinely work in IDL, Python, Matlab, etc.
    or a post-doc / motivated grad student)
  - easy to write **clients** from scratch as well
- REST-like flavor: specific URL endpoints, each with one service
- Other constraints
  - designed with tabular, time-series data in mind
    - think: records of data, where each record has a time stamp and a set of parameters
  - initial focus was data up to 2-D data (spectrogram, pitch angle distributions, etc)
  - current version now supports arbitrary sizes for arrays

Details of the HAPI specification (focusing on minimum requirements)

The 4 required HAPI Endpoints:

All endpoints must be directly below a URL that ends with '`hapi`'

- `http://example.com/hapi/capabilities`
  - describes options implemented by the server
- `http://example.com/hapi/catalog`
  - list of datasets at the server
- `http://example.com/hapi/info`
  - show metadata for one dataset at a time (basically a data header)
- `http://example.com/hapi/data`
  - retrieve a stream of data content for one dataset over a specific time range

The `capabilities` endpoint

`http://example.com/hapi/capabilities`

- response is in JSON format
- all outputs include the API version number
- `capabilities` endpoint indicates which optional elements of the serer are implemented by this server; currently, there is only one such option, namely the formats the server can output
- Data output formats: a server MUST support `csv` , while `binary and json are optional.`
- Example:

```
{
   "HAPI" : "1.1",
   "capabilities" :
   [
      {
        "formats": ["csv", "binary","json" ]
      }
   ]
}
```

The `catalog` endpoint

`http://example.com/hapi/catalog`

- response is JSON again
- list of just the identifiers of the datasets available at this server
- no metadata for the datasets, just the id
- Example:

```
{
   "HAPI" : "1.1",
   "catalog" :
   [
      {"id": "path/to/ACE_MAG" "title": "ACE Mag Data"},
      {"id": "data/IBEX/ENA/AVG5MIN"},
      {"id": "data/CRUISE/PLS"},

      {"id": "any_identifier_here"
       "title": "optional human readable plot label" }
   ]
}
```

The `info` endpoint

`http://example.com/hapi/info?id=DATASET_ID`

- **The request:**
  - one require HTTP request parameter `id` indicating the dataset
  - there is an optional input to limit which parameters are described
    - `hapi/info?id=DATASET_ID&parameters=A,B,C,D`

- **The response**
  - JSON formatted metadata about the requested dataset
  - the focus is the list of parameters in the dataset
  - a few required elements in the metadata, and several optional elements

(Example on the next slide…)

`info` endpoint example

Request: `http://example.com/hapi/info?id=path/to/ACE/MAG`

```
{   "HAPI": "1.1",
    "creationDate”: "2016-06-15T12:34”
    "startDate": "1997-08-01T12:00",
    "stopDate": "2017-005",
    "parameters": [
        { "name": "Time",
          "type": "isotime",
          "length": 24,
          "fill": null },
        { "name": "quality flag",
          "type": "integer”,
          "description ": "0=OK and 1=bad" },
          "fill": null },
        { "name": "mag_GSE",
          "type": "float",
          "units": "nT",
          "size" : [3],
          "fill": -99999.999,
          "description": "hr avg Cartesian mag field in nT in GSE" }
    ]
}
```

NOTES:
- *dataset name can have slashes;*
- *FILL value **must** be specified for for each parameter, or you must explicitly indicate that there is no fill*

HAPI dataset `parameters`

Attribute	Type	Description
`name`	string	**Required**
`type`	string	**Required**; One of "string", "double", "integer", "isotime"
`length`	integer	**Required for 'string' and 'isotime'**; not allowed for others
`units`	string	**Optional**. Default is 'dimensionless' for everything but 'isotime' types
`size`	array(Integer)	**Required for array parameters**; not allowed for others; number of elements in each dimension
`fill`	string	**Required**. The fill value for this parameter; lots more details since it is a string
`description`	string	**Optional**. A brief description of the parameter
`bins`	object	**Optional**. For array parameters, the bins object describes the values associated with each element in the array.

only 'name' and 'type' and 'fill' are required for everything

Output Time Values

- Any ISO 8601 string
- can leave off date elements if they are zero (or 1 for day of year or month day)
- can optionally end with “Z”
- servers allowed to emit either YYYY-DDD or YYYY-MM-DD, so clients must be able to handle both date flavors

- even in binary output, time values are still strings (must specify a fixed length)

- all time values in a dataset should be the same (don’t omit time fields on just some values – be consistent)

The `data` endpoint

```
http://example.com/hapi/data?id=DATASET_ID&
        time.min=2016-01-01T12:34:56.789Z&
        time.max=2016-01-01T18
```

- dataset id is required
- time range is required
  - time format can be any valid ISO 8601 time string (day-of-year or year-month-day) and all time fields need not be specified (just the date is OK, or just the date and the hour)
- parameter restriction is optional
  - `hapi/data?id=DATASET_ID&time.min=T0&time.max=T1&`

    `parameters=A,B,C,D`
- default format is 'csv'; 'binary' or 'json' can be specified (if supported in the capabilities)
- dataset header (from the 'info' request) can be prepended to the data and each line will be prefixed with a "#" character
- Example on the next slide

data endpoint example

```
http://datashop.elasticbeanstalk.com/hapi/data?
  id=spase://VSPO/NumericalData/Cassini/MAG/PT60S
  &time.min=2002-095&time.max=2002-095T00:05
  &include=header&parameters=BR,BT,BN
```

```
#{ "HAPI": "1.0",
#  "creationDate": "2016-10-13T18:40:20.000",
#  "parameters": [
#    { "name": "Epoch", "type": "isotime", "units": "UTC", "length": 23,"fill":null },
#    { "name": "BR", "type": "double", "units": "nT", "fill": "-1.000e+38" },
#    { "name": "BT", "type": "double", "units": "nT", "fill": "-1.000e+38" },
#    { "name": "BN", "type": "double", "units": "nT", "fill": "-1.000e+38" }],
#  "startDate": "2000-01-01T00:00:00.000",
#  "stopDate": "2004-07-01T00:00:00.000",
#  "description": "Cassini cruise magnetometer data at 1 min resolution",
#  "resourceURL": "https://spdf.gsfc.nasa.gov/pub/data/cassini/",
#  "resourceID": "spase://VSPO/NumericalData/Cassini/MAG/PT60S",
#  "creationDate": "2017-04-02T01:46:33.000",
#  "cadence": "PT1M",
#  "format": "csv"
#}
2002-04-05T00:00:00.000,2.040000e-01,-6.300000e-01,6.700000e-02
2002-04-05T00:01:00.000,1.880000e-01,-6.340000e-01,7.100000e-02
2002-04-05T00:02:00.000,1.660000e-01,-6.390000e-01,6.200000e-02
2002-04-05T00:03:00.000,1.770000e-01,-6.410000e-01,6.500000e-02
2002-04-05T00:04:00.000,1.990000e-01,-6.300000e-01,4.400000e-02
2002-04-05T00:05:00.000,2.010000e-01,-6.280000e-01,3.200000e-02
```

Example HAPI Servers

- These are still prototypes – they are working, but have not been tested for capacity loading, robustness, etc. .They are there to test out ideas during the ongoing development of the spec.

- GSFC

  http://cdaweb.gsfc.nasa.gov/registry/hdp/hapi/

- JHU / APL

  http://datashop.elasticbeanstalk.com/hapi/

- George Mason University

  http://tsds.org/get/SSCWeb/hapi

- Autoplot exploration server

  http://jfaden.net/HapiServerDemo/hapi

HAPI Clients

- **Autoplot can read from a HAPI server**
  - **NOTE: You need the development version of Autoplot and you need the line**

```
"hapiDeveloper=true"
```

in ~/autoplot_data/config/system.properties for this to work.

- **An IDL client written by Scott Boardsen at GSFC**

Demonstrations

- **Server**

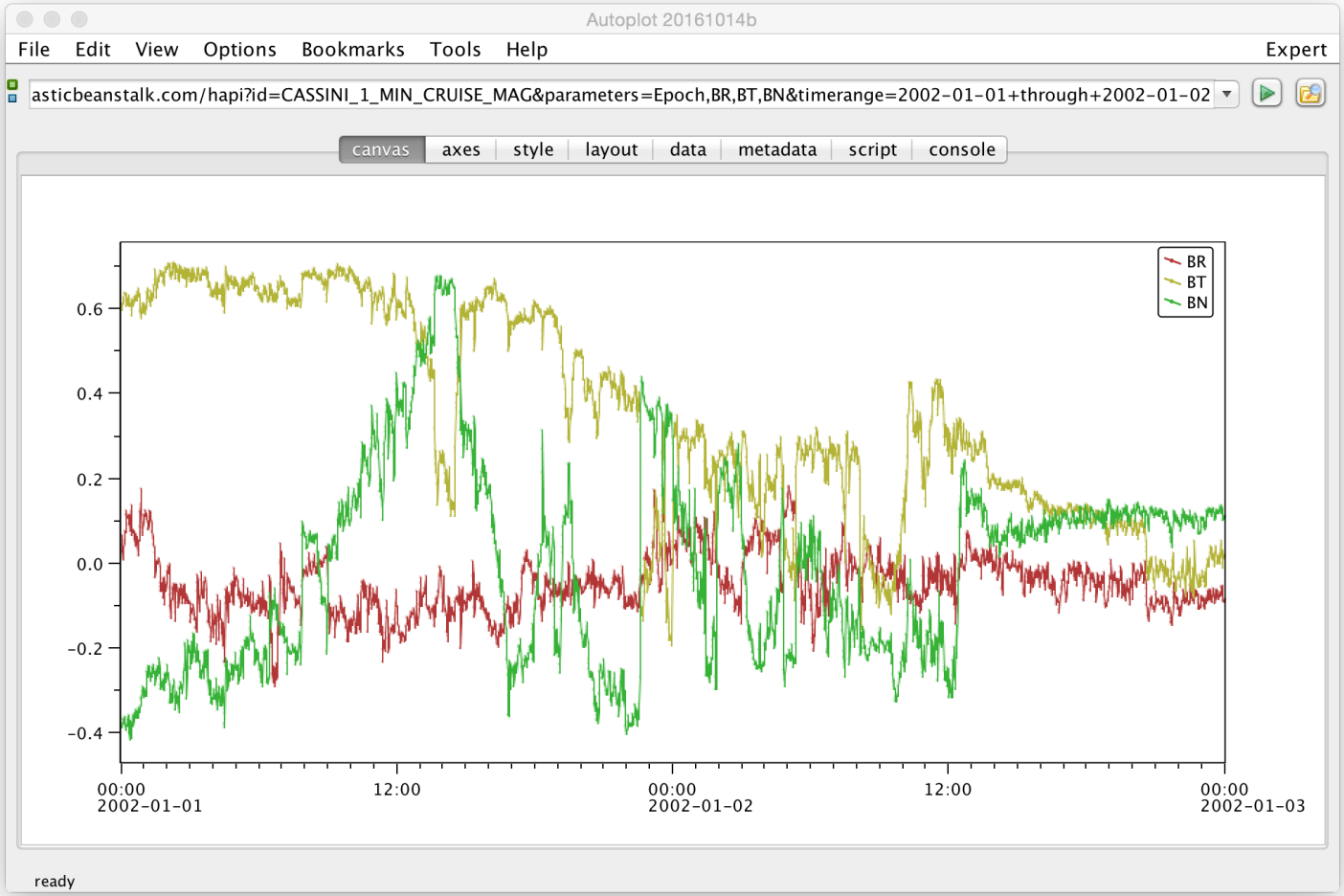
  http://datashop.elasticbeanstalk.com/hapi/

- **Client**

  autplot.org (development version with hapiDeveloper=true)

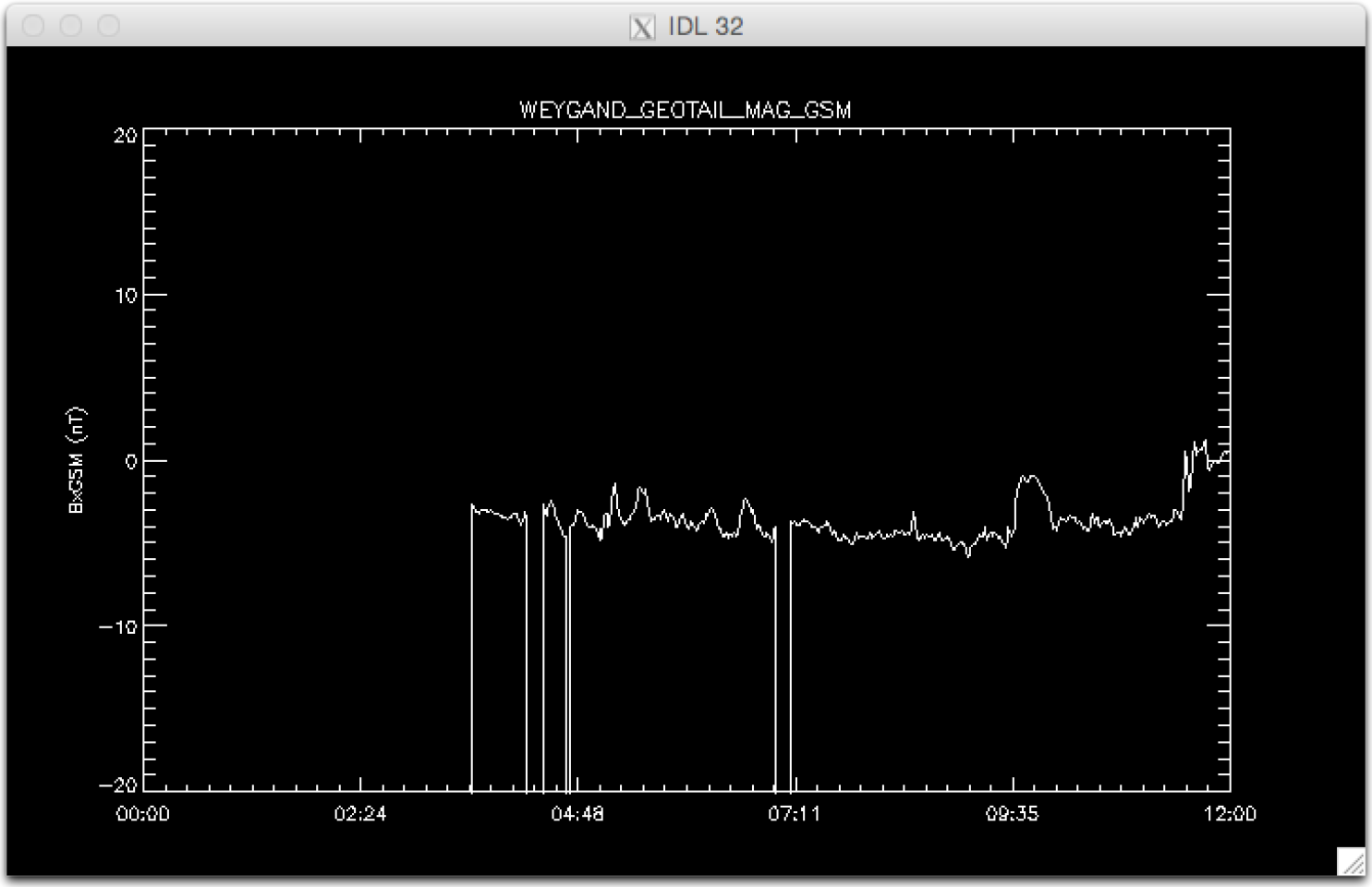
Autoplot

```
vap+hapi:http://datashop.elasticbeanstalk.com/hapi?
id=CASSINI_1_MIN_CRUISE_MAG&parameters=Epoch,BR,BT,BN&
timerange=2002-01-01+through+2002-01-02
```



IDL Client (Boardsen)

Plot of Geotail MAG data from a UCLA server.



Standardizing on the API rather than the format

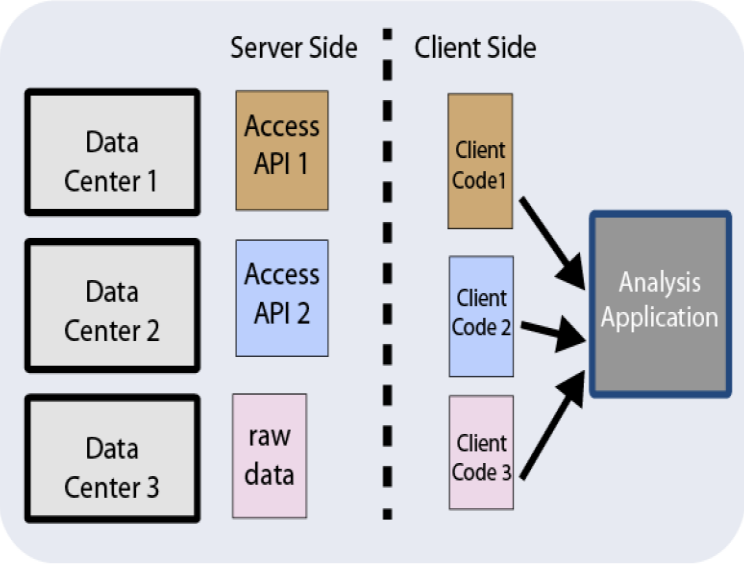
Past

Different data formats make interoperability difficult. ASCII (many flavors), CDF, HDF4, HDF5, netCDF, CEF, custom binary, etc.

There has been considerable progress in the use of standard formats recently, but there will never be just one data format.

Present

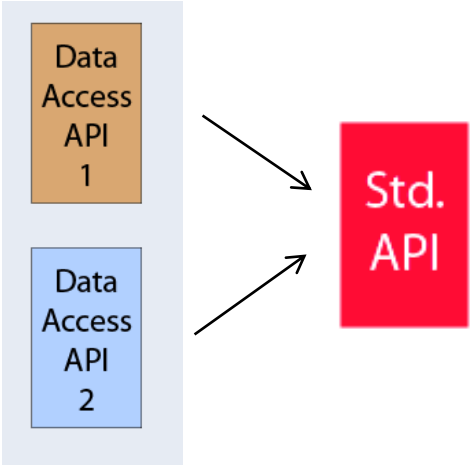
Data servers have consolidated many datasets, but each data center now has a different access interface.



Three access modes are shown here. The first two data centers have their own distinct APIs and the third data center just exposes a directory structure of files.

Future

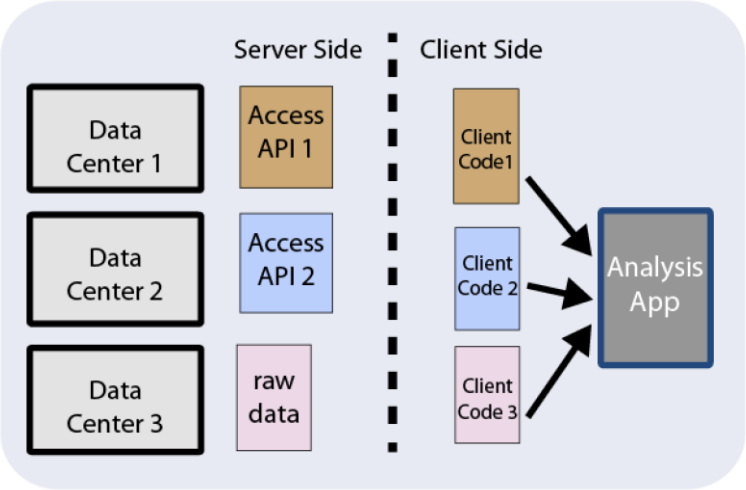
need a domain-specific, but standardized API representing a lowest common denominator that multiple data centers will endorse and implement



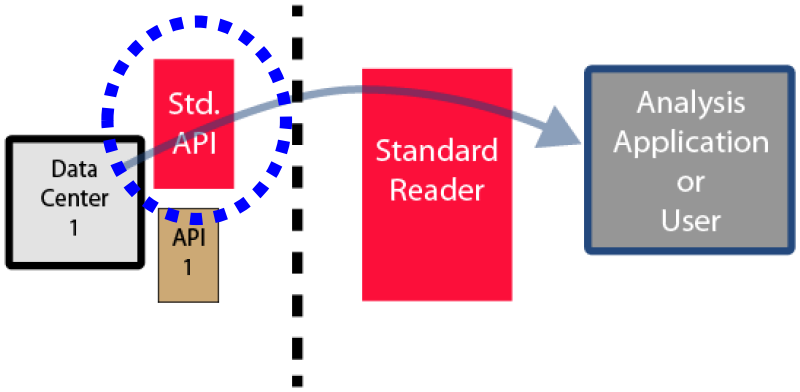
note: also need to incorporate data centers without any existing API

Scenarios for Using a Standard Data Server API
Case 1 of 3

Reminder: each data center has its own API



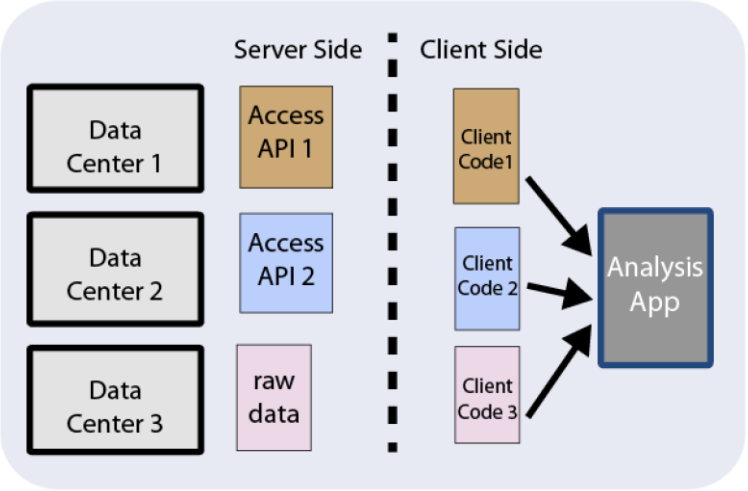
Data Center creates a new access mechanism to support the new interface.



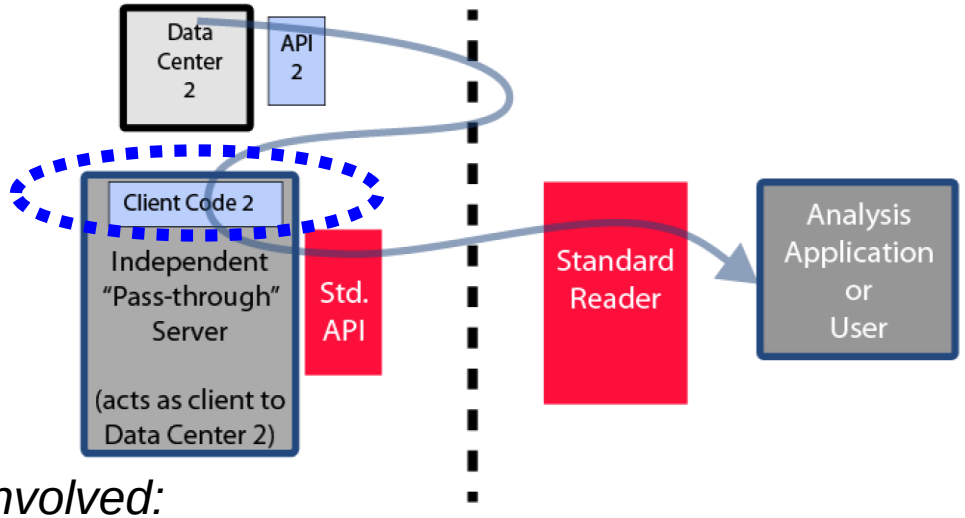
main effort involved:
server implementation created (or used) and maintained by Data Center 1
(the blue, dashed circle)

Scenarios for a Standard Data Server API
Case 2 of 3

Reminder: each data center has its own API



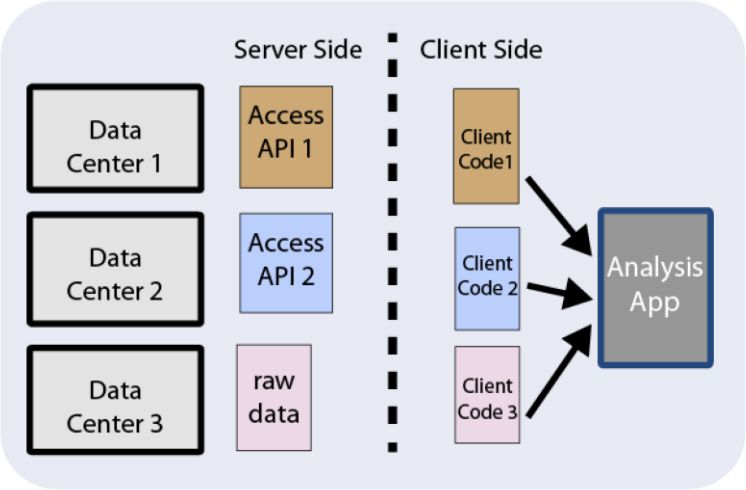
A Data Center (large gray box) serves as a pass-through server by creating an adapter to read from the non-standard API of another data center (small light gray box) and re-serve the data using a standard API.



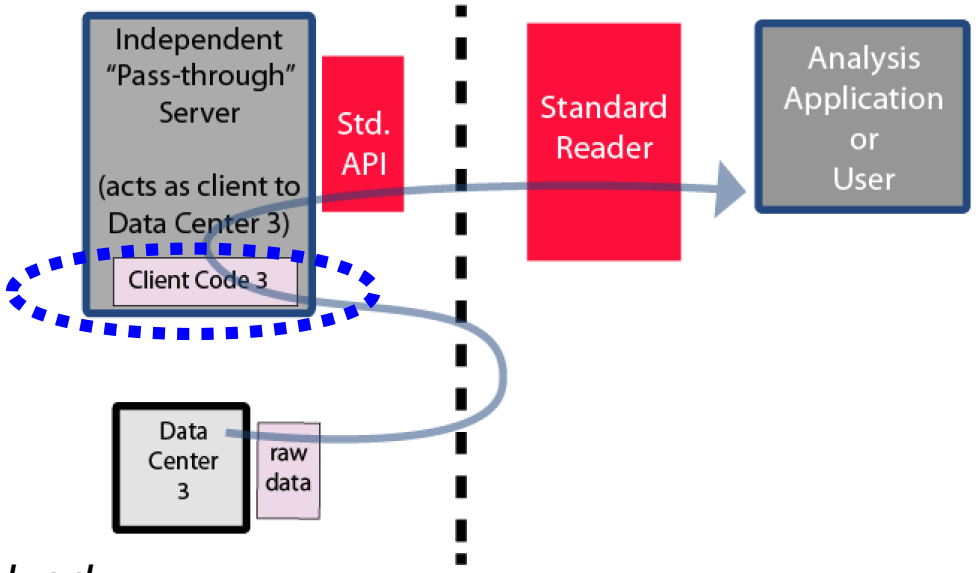
main effort involved:
reader for API 2 written and maintained by Data Center 2, who also has their own HAPI server
(the blue, dashed oval)

Scenarios for a Standard Data Server API
Case 3 of 3

Reminder: each data center has its own API



Data Center creates a new access mechanism to support the new interface.



main effort involved:

reader for raw data written and maintained by Data Center 2, who also has their own HAPI server

(the blue, dashed oval)

HAPI and SPASE – designed for different uses

- **SPASE**
  - captures metadata in a standard way and enables interoperable searches for resources, but has limited use for allowing access to the data – mostly by design (e.g., not too many required elements, no logical link between SPASE and data files)
  - very mature; many datasets described
  - used by Heliophysics Data Portal, VMO, VHO, VEPO and others internationally

- **HAPI – Heliophysics Application Programming Interface**
  - focus is access to the raw numbers
  - designed to be somewhat independent of SPASE and to be focused on access to the raw numbers in the data files
  - very new: version 1.1 spec is pretty much ready
  - a few demonstration servers and test clients
  - buy-in or interest from many organizations:
    - GSFC / CDAWeb, PDS / PPI Node, GMU, U Iowa / Autoplot, JHU/APL, LASP, CCMC, European planetary efforts

Using the HAPI Specification

The specification is online:

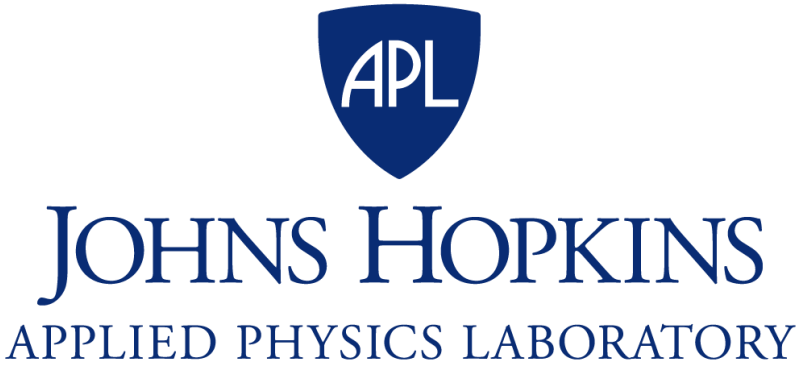
`https://github.com/hapi-server/data-specification`

Version 1.1 is under development and will be released soon.

The spec alone is enough to implement a server or client, but talk to us – we are very interested in working directly with people in the early stages of HAPI adoption.

Future Plans

- **HDMC will support the creation of reference implementations for A HAPI server that will function as**
  - **an example of how to expose existing data via the HAPI Spec**
  - **a starting point for a more turnkey solution: a short interview about your existing data will configure a ready-to-go-server. These will help:**
    - **data are in a self-describing data format**
    - **file names follow a rational date-based convention**
    - **metadata is available somewhere, hopefully in high quality SPASE**
    - **If these are true, the turnkey solution will work.**
- **Also, add elements to space to make this easier to adapt directly from parameter level SPASE**
  - **existing <accessUrl/> element can point to a HAPI server**
  - **add: URI template string (describes date elements in filenames)**
  - **add: units for each parameter**


APL
JOHNS HOPKINS
APPLIED PHYSICS LABORATORY